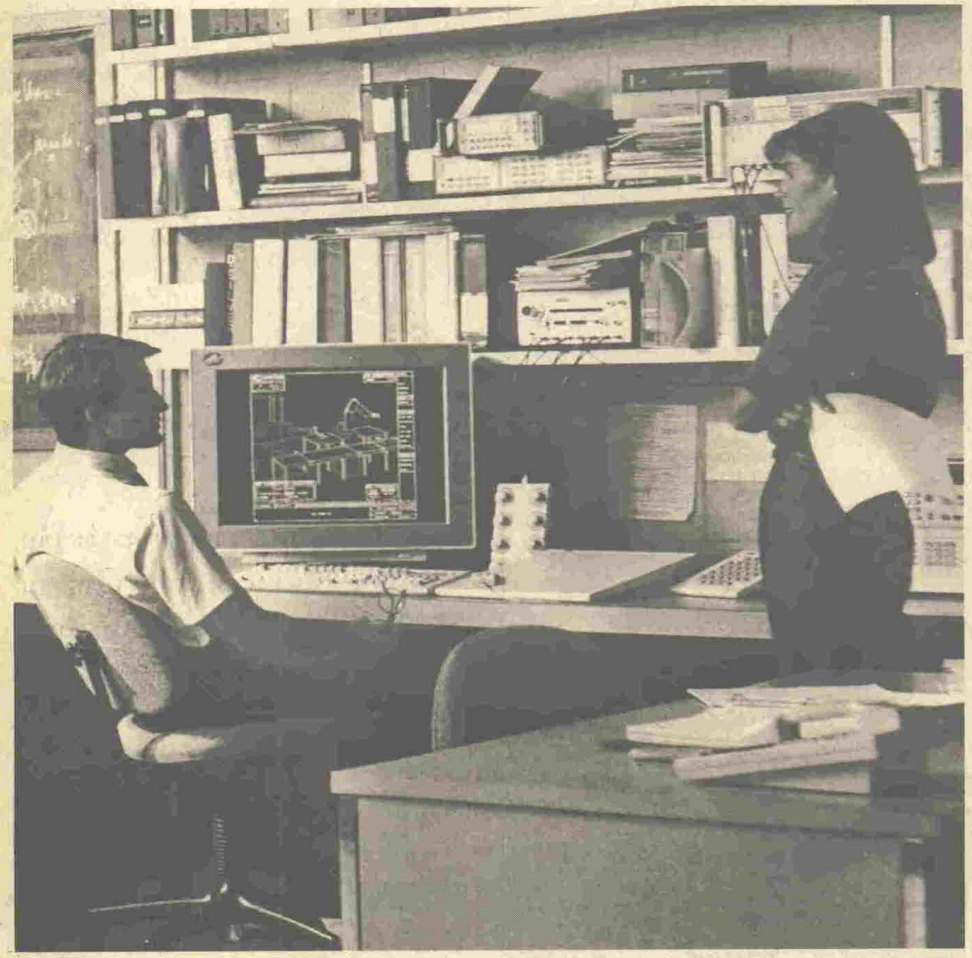
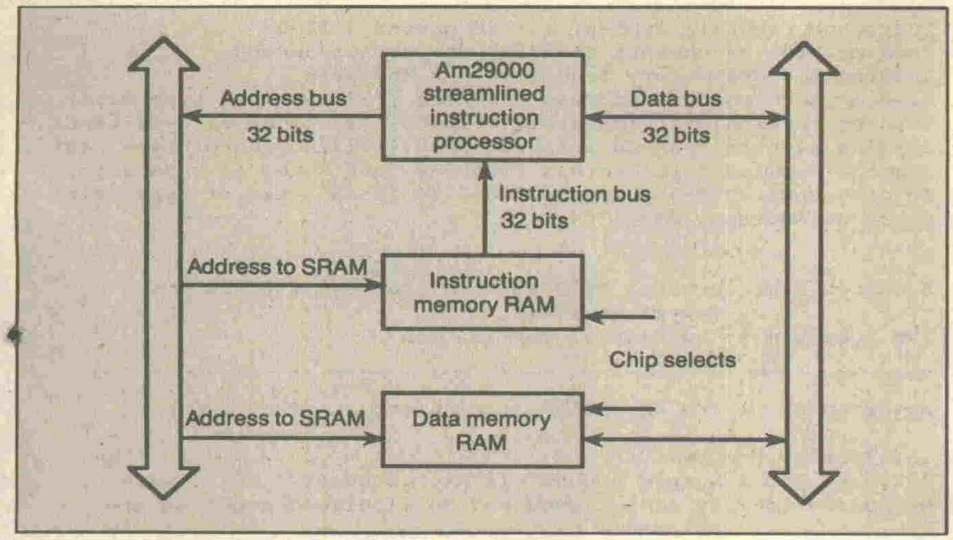
เอนก ศิรจริยาภิวัฒน์

RISC
ซีพียูแห่งอนาคต

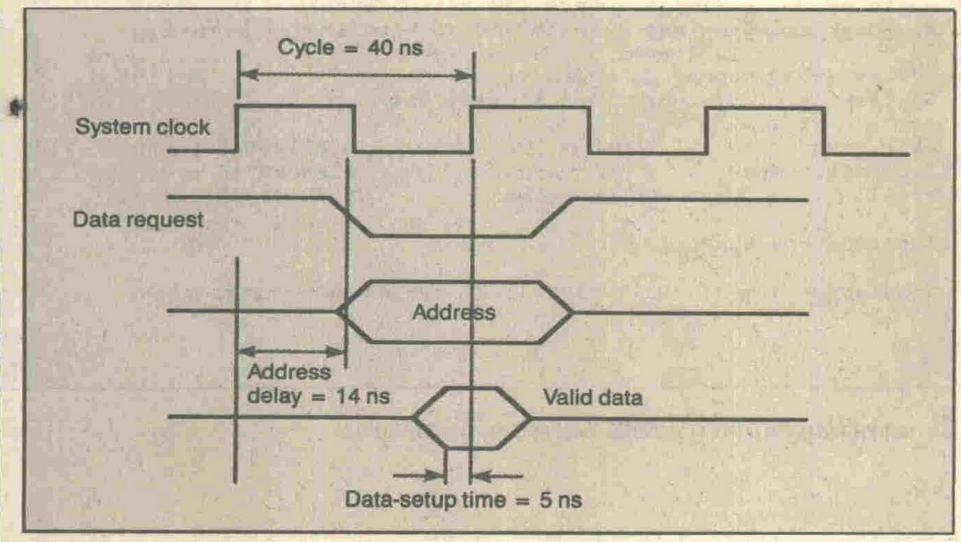
จากความก้าวหน้าทางคอมพิวเตอร์กราฟิกอิเมจเป็นสิ่งหนึ่งที่ก้าวตามไปด้วย RISC จึงเป็นทางออกของทั้งสองเรื่อง แต่....



ส่วนมากเมื่อมีการพูดถึง CPU เราจะคำนึงถึงเฉพาะตัวของ CPU โดยไม่ได้นึกถึงส่วนสำคัญอีกส่วนหนึ่งก็คือหน่วยความจำ วันนี้เราจะมาพูดถึงความสัมพันธ์และการทำงานของ CPU กับหน่วยความจำ ชนิดของหน่วยความจำและความเร็วของหน่วยความจำไม่ได้เป็นตัววัดความสามารถและคุณสมบัติของเครื่องคอมพิวเตอร์ที่สำคัญเมื่อเทียบการออกแบบและติดตั้งส่วนเชื่อมต่อ (interface) กับ CPU ซึ่งกลายเป็นกุญแจสำคัญ การที่เราจะได้ประโยชน์สูงสุดจาก CPU ที่มีความเร็วสูงมาก (ultrafast) นั้น ฮาร์ดแวร์รอบด้าน (RAM, ROM และ peripheral) จำเป็นที่จะต้องทำงานให้ทันในช่วงเวลาสัญญาณนาฬิกาของ CPU นั้น กรณีของ Reduced instruction set computer (RISC) processor ขนาด 25 MHz เช่น AM29000 ตัว CPU จะต้องได้รับข้อมูลขนาด 32 bit word ในช่วงระยะเวลา 40 นาโนวินาที การเชื่อมต่อระหว่างหน่วยความจำกับ CPU นี้มีการพัฒนาจนกลายเป็นประวัติศาสตร์ของการนำเสนอเทคนิคใหม่ ๆ มากมายในการออกแบบที่กำหนดข้อมูลที่จะมีให้ CPU ทันในช่วงระยะเวลา 40 นาโนวินาที ขอยกตัวอย่างการเชื่อมของ Static RAM (SRAM) ดังรูปที่ 1, 2 เพื่อให้ได้ข้อมูลภายในเวลา 40 นาโนวินาที เราจำเป็นที่จะต้องได้เวลาการเข้าถึง (access time) ที่มีค่าอย่างน้อย



รูปที่ 1 การต่อ SRAM แบบ zero wait state สำหรับ AM29000 สังเกตว่าบัสคำสั่งและบัสข้อมูลแยกกัน แต่บัสของแอดเดรสใช้รวมกัน



รูปที่ 2 ลักษณะของเวลา (timing diagram) สำหรับการออกแบบในรูปที่ 1

21 นาโนวินาที ซึ่งก็คือ เวลาใน 1 ลูกสัญญาณนาฬิกา (40 ns)–เวลาหน่วงของสัญญาณแอดเดรส (14 ns)–เวลาที่ข้อมูลจะถือว่าใช้ได้ (data setup time 5 ns) ซึ่งเวลานี้ยังไม่ได้รวมถึงเวลาที่เสียไปกับบัฟเฟอร์และส่วนควบคุมต่าง ๆ เราจะพบเวลาการเข้าถึงขนาด 20 นาโนวินาทีนี้ไม่เป็นจริงในทางปฏิบัติ จึงมาถึงคำถามที่ว่าทำไมบริษัทต่าง ๆ จึงผลิต CPU ที่มีความเร็วสูงขึ้นเรื่อย ๆ จนไม่สามารถทำงานได้ที่ความเร็วสูงสุดด้วย RAM ที่พอจะหาได้ปัจจุบัน คำตอบอยู่ที่การคำนวณเวลาการเข้าถึงหน่วยความจำและเทคนิคการออกแบบการเชื่อมต่อซึ่งเทคนิคต่าง ๆ ได้เข้ามาสู่เครื่องไมโครคอมพิวเตอร์เมื่อ 20 ปีก่อน ไมโครโปรเซสเซอร์ในรุ่นหลังเช่น AM 29000, MC88000 หรือ 80386 จะใช้เทคนิคของ interleaving, pipelining และ burst mode เพื่อให้ได้การทำงานด้วยประสิทธิภาพสูงสุด

การจำลองแบบของ RISCs

เพื่อเป็นการทดสอบความสามารถและประสิทธิภาพ การเชื่อมต่อในหลาย ๆ เทคนิค โดยไม่ต้องทำการประกอบวงจรจริง ทางบริษัท AMD ได้ให้โปรแกรมจำลองการทำงานของ AM29000 โดยเป็นซอฟต์แวร์ที่ทำงานภายใต้ MS-DOS ส่วน execution unit ของ AM29000 จะใช้ four-stage pipeline ซึ่งทำให้การทำงานสูงสุด 1 คำสั่งทุก ๆ สัญญาณนาฬิกา (40 ns) ตัว AM 29000 มีขนาด 32 บิต คือ แอดเดรสบัส ดาต้าบัส อินสตรัคชันบัส ดังรูปที่ 1 โดยเป็นบัสแบบ non multiplex โดยแยกส่วนของข้อมูลและคำสั่งออกจากกันเป็น 2 บัส แต่ใช้แอดเดรสบัสร่วมกัน โปรแกรมจำลองการทำงานของ AM29000 จะบอกถึงการออกแบบด้วยเงื่อนไขดังที่กล่าวต่อไปนี้

ไม่มีการรอ (No Waiting)

การออกแบบหน่วยความจำดังรูปที่ 1 โดยไม่มีการรอคอย (Zero-wait-state) นั้น เมื่อเราใช้โปรแกรมจำลองการทำงานของ AM29000 เราจะได้ผลดังรูปที่ 3 ซึ่งโปรแกรมจำลองการทำงานของ AM29000 บอกกับเราว่า CPU จะสามารถทำงานได้ในระดับ 20.71 ล้านคำสั่งต่อวินาที (MIPS) และ 39,698 Dhrystones ต่อวินาที ซึ่งจะเห็นว่ามีประสิทธิภาพแค่ 94.7% ของค่าสูงสุดที่ควรจะเป็นเท่านั้น

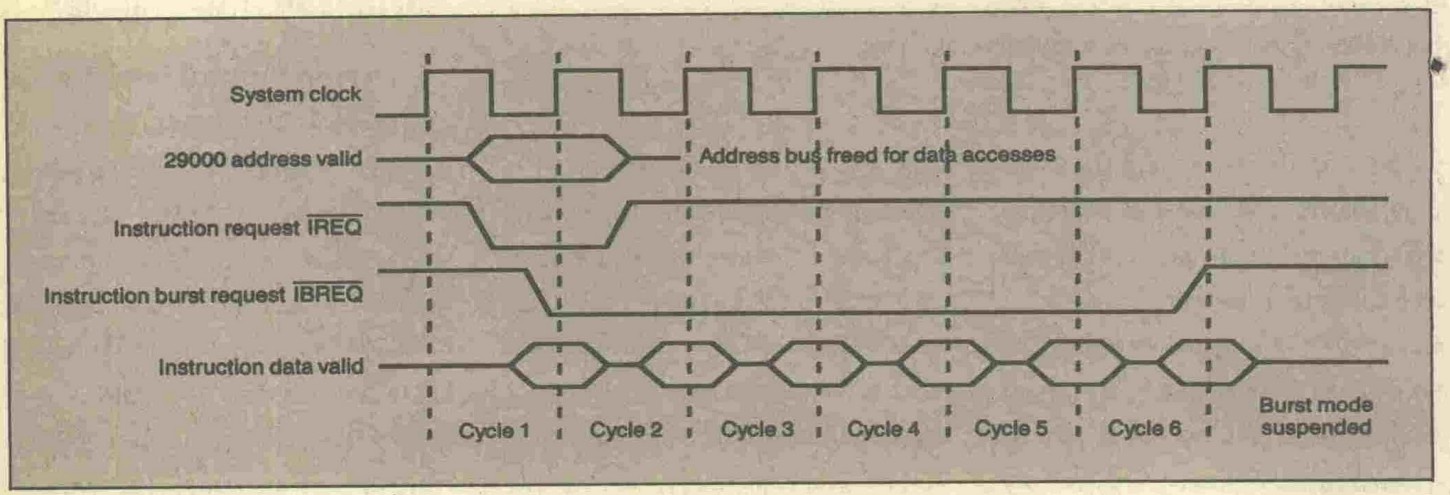
การมาเป็นขบวน

(Bursting Through)

ในการที่เราจะเข้าใจว่าอะไรเกิดขึ้นเมื่อเราดูการทำงานของ CPU ทั้ง 4 ขั้นตอน (four-stage execution pipeline operaters) เราจะคิดถึงการนำคำสั่งเข้าตีความ (Instruction fetches) นั้นมีการเลื่อมกับการนำเข้าข้อมูล (data fetches) ซึ่งเกิดขึ้นได้แม้ว่าบัสของข้อมูลและบัสของคำสั่งจะแยกกัน อย่างไรก็ตามทั้งชุดคำสั่งและข้อมูลก็สามารถร่วมกันใช้แอดเดรสได้ แต่เมื่อทั้งคู่ต้องการบัสของแอดเดรสในเวลาเดียวกันก็จะเกิดปัญหาขึ้น คำตอบอยู่ที่การทำงานแบบ Brust mode โดยจะใช้แอดเดรสบัสเพียงครั้งเดียวในการนำข้อมูลหนึ่งชุดไปใช้งาน (ใช้ค่าแอดเดรสเพียงแอดเดรสเดียว) ในรูปที่ 4 จะเป็นการทำงานของ Burst mode เพื่อใช้ในการนำคำสั่งเข้าตีความ โดยเริ่มจากการนำแอดเดรสตัวแรกของข้อมูลไปวางบนแอดเดรสบัส ในช่วงไซเคิลแรกซึ่งจะมีผลให้ส่วนควบคุมฮาร์ดแวร์ของ RAM เพิ่มค่าแอดเดรสขึ้นทุก ๆ ไซเคิล เมื่อสัญญาณ Burst-request (IBRQ) ทำ

```
Dhrystone time (in cycles) for 50 passes = 31487
This machine benchmarks at 39698 dhrystones/second
Loading Am29000 Memory from file: mw_dhry.bin.
loading section " at address 00002000 [228 bytes of type data]
loading section " at address 00001000 [1924 bytes of type text]
loading section " at address 00003000 [10632 bytes of type bss]
loading section " at address 00000000 [564 bytes of type data]
loading section " at address 00001784 [1568 bytes of type text]
Entry at Address: 00001784


Advanced Micro Devices Am29000 Simulator Ver 4.21-PC -
              Copyright 1987
Sim complete -- successful termination

_____________________________________________________

Environment of "mw_dhry.bin" simulation:

Instruction Memory:
1 Cycles for a Simple access. (0 Wait States)
No Burst accesses are allowed and no Pipelined accesses are
              allowed.
(0 Cycles To Decode an Address)
Instruction ROM Memory:
1 Cycles for a Simple access. 0 Cycles To Decode an address.

Data Memory:
1 Cycles for a Simple access. (0 Wait States)
No Burst accesses are allowed and no Pipelined accesses are
              allowed.
(0 Cycles To Decode an Address)
Statistics of "mw_dhry.bin" simulation:

User Mode:          32404 cycles          (0.00129616 seconds)
Supervisor Mode:      189 cycles          (0.00000756 seconds)
Total:              32593 cycles          (0.00130372 seconds)

Instructions Executed:          27006

Simulation speed: 20.71 MIPS (1.21 cycles per instruction)
```

รูปที่ 3 ผลการจำลองการทำงานของ AM29000 โดยออกแบบฮาร์ดแวร์ดังรูปที่ 1



รูปที่ 4 ลักษณะการทำ Burst mode

```
---------- Pipeline ----------
 17.14% idle pipeline:
      12.70% Instruction Fetch Wait
       2.50% Data Transaction Wait
       0.00% Page Boundary Crossing Fetch Wait
       0.03% Unfilled Cache Fetch Wait
       0.00% Load/Store Multiple Executing
       1.54% Load/Load Transaction Wait
       0.38% Pipeline Latency

---------- Branch Target Cache ----------
Branch cache access:            13511
Branch cache hits:               8197
Branch cache hit ratio:        60.67%

---------- Translation Lookaside Buffer ----------
TLB access:                      9433
TLB hits:                        9426
TLB hit ratio:                 99.93%

---------- Bus Utilization ----------
Inst Bus Utilization:          71.44%
       23285 Instruction Fetches

Data Bus Utilization:          18.11%
        3380 Loads
        2523 Stores

---------- Instruction Mix ----------

       3.24% Calls
      14.41% Jumps
      12.52% Loads
       9.34% Stores
       6.49% No-ops

---------- Register File Spilling/Filling ----------
           0 Spills
           0 Fills


(Simulator Performance:      592.60 cycles per second)
```

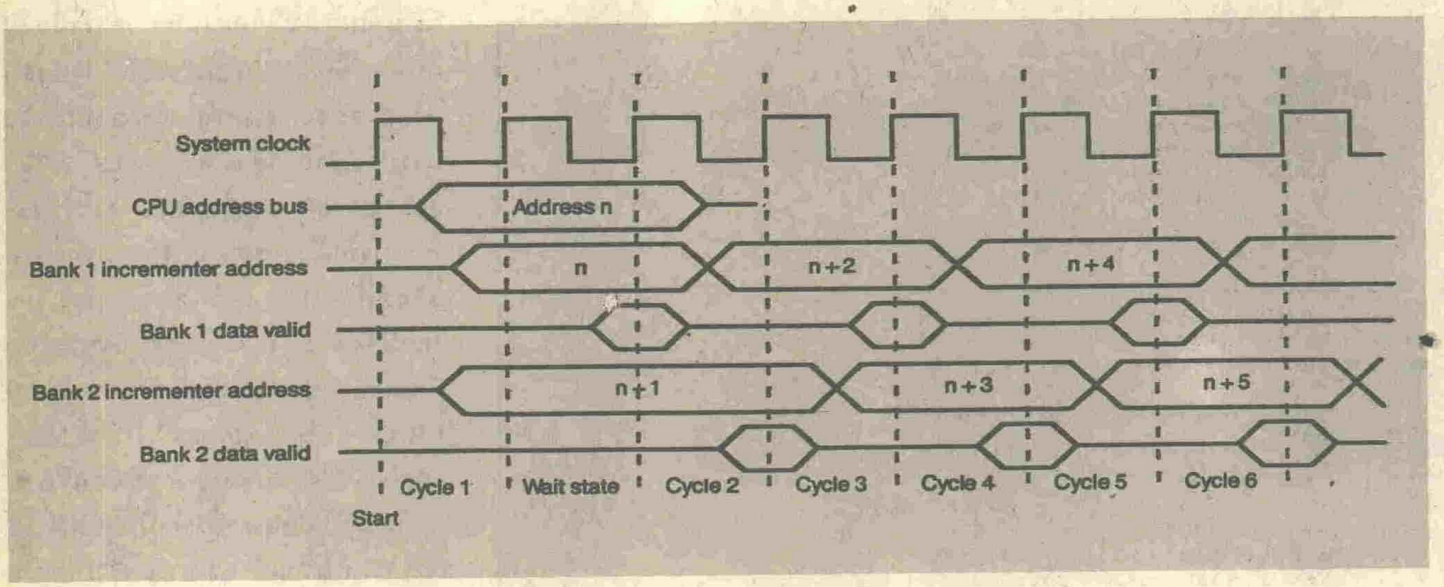
งาน อันจะทำให้บัสแอดเดรสว่างลง เพื่อจะใช้นำข้อมูลเข้า (data fetches) คำสั่งส่วนมากจะเรียงลำดับกันมาอยู่แล้ว แต่เราก็จะเจอปัญหากับคำสั่ง BRANCH ซึ่งทำให้เกิดทำงานในส่วนที่ไม่ได้เรียงต่อกันมา ชุดคำสั่งนั้นก็จะถูกอินเตอร์รัพต์ไป ในตัว AM29000 จะมี Branch-target cache เพื่อเก็บคำสั่งไม่เกิน 4 คำสั่ง และจะถูกใช้ต่อจากคำสั่ง Branch ในเมมโมรีภายใน CPU (Cache) ซึ่งถ้ามีการเรียกคำสั่งชุดนี้อีกก็สามารถดึงได้จาก Cache ไม่ต้องรบกวนเมนเมมโมรีอีก

การหยุดรอหน่วยความจำ (Waiting for memory)

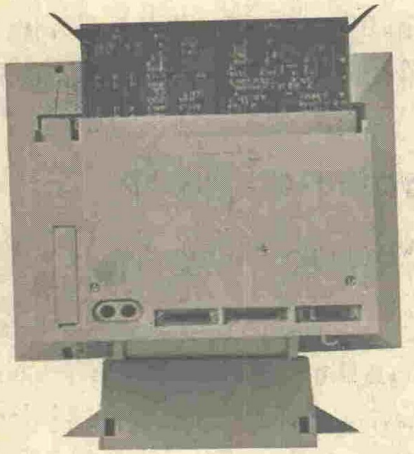
หน่วยความจำส่วนใหญ่ในทางปฏิบัติจะมีเวลาการเข้าถึง (access time) มากกว่า 20 นาโนวินาที ซึ่งถ้าเราจำลองการทำงานในรูปที่ 1 โดยมีจังหวะการรอคอย 1 จังหวะ ช่วยสัญญาณนาฬิกา 80 นาโนวินาที, แอดเดรสดีเลย์เป็น 14 นาโนวินาที และดาต้าเซตอินเป็น 5 นาโนวินาทีจะได้ความเร็วเฉลี่ย 61 นาโนวินาที โดยไม่คิดเวลาที่เสียไปกับส่วนอื่น ๆ เช่น บัฟเฟอร์ โปรแกรมจำลองจะรายงานผลของความเร็วในการประมวลคำสั่งเป็น 14.08 MIPS และ 26,907 Dhrystones ซึ่งใช้ประสิทธิภาพเพียง 64% ของความสามารถสูงสุดที่ควรเป็น ถ้าจำลองการทำงานโดยมีจังหวะการรอคอย 3 จังหวะ ซึ่งเป็นค่าประมาณของหน่วยความจำแบบ DRAM จะได้ผลออกมาเป็น 7.42 MIPS และ 14,104 Dhrystones ประสิทธิภาพลดลงมาเป็น 34% ซึ่งจะเป็นจุดที่ทำให้เรามองเห็นว่าแม้ CPU จะเร็วเพียงใด แต่ถ้าไม่มีระบบหน่วยความจำที่มีความเร็วสูงเพียงพอ ก็จะทำให้ CPU นั้นไม่สามารถใช้ความสามารถสูงสุดได้ และนี่คือสาเหตุที่เครื่องคอมพิวเตอร์ที่ใช้ RISCs ไปไม่ถึงจุดที่ตั้งใจไว้สักที

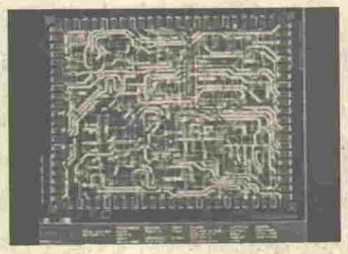
เพิ่มความเร็วขึ้นอีก (Speeding Things Up)

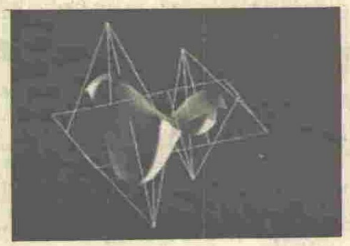
เทคนิคที่เราพูดผ่านมาก็คือ Burst mode ต่อไปเราจะพูดถึงการทำ interleaving ซึ่งใช้หน่วยความจำ 2 ส่วน (Blank) แทนการใช้หน่วยความจำส่วนเดียว โดยส่วนหนึ่งใช้สำหรับแอดเดรสที่เป็นเลขคี่ อีกส่วนสำหรับแอดเดรสที่เป็นเลขคู่ เราจะสมมติว่าคำสั่งเข้ามาเป็นลำดับทำให้หน่วยความจำส่วนหนึ่งทำงาน จนกระทั่งได้รับสัญญาณ row-address-strobe (RAS) สำหรับ DRAM หรือแอดเดรสต่อไป ในกรณี SRAM จะทำให้หน่วยความจำอีกส่วนทำงาน เรามาดูการทำงานแบบนี้ในรูปที่ 5 โดย AM29000 ถ้าเกิดเราออกแบบให้มี 1 จังหวะการรอคอย จะทำให้เราเริ่มเข้าถึงหน่วยความจำส่วนแรกในความเร็ว



รูปที่ 5 ลักษณะการทำ interleaved







60 นาโนวินาที และหน่วยความจำส่วนที่สองในเวลา 100 นาโนวินาที แต่เมื่อทำงานในช่วงต่อไปจะทำให้หน่วยความจำส่วนที่ 1 และส่วนที่ 2 มีข้อมูลของตัวเองได้ในระยะเวลา 40 นาโนวินาที ดังนั้น interleaved burst access จะใช้ 1 จังหวะการรอคอยในการเริ่มทำงานต่อไปก็จะเป็น zero wait state ซึ่งผลของการจำลองการทำงานด้วยวิธีนี้ทำให้เราได้ผลออกมา 18.64 MIPS และ 35,760 Dhrystones (85% ของค่าสูงสุด) เป็นการเพิ่มประสิทธิภาพจาก burst mode ธรรมดาขึ้น 21% เพื่อรวมผลกระทบรอบข้าง เช่น idle time จะทำให้ประสิทธิภาพของวิธีนี้น้อยกว่าแบบ zero wait state อยู่แค่ 2.8%

Pipeling

การทำ pipeling นั้นแอดเดรสของคำสั่งต่อไปจะถูกส่งให้กับบัสของแอดเดรสก่อนที่จะจบไซเคิลการทำงานปัจจุบัน โดยจะมีฮาร์ดแวร์ภายนอกค้าง (Lactch) ค่าของแอดเดรสนั้นไว้ให้ ซึ่งจะทำให้ประสิทธิภาพการทำงานของระบบเพิ่มขึ้นประมาณ 2.6% ในปัจจุบันนอกจากเทคนิคต่าง ๆ ที่กล่าวมาหน่วยความจำของเครื่องคอมพิวเตอร์บางเครื่องจะใช้ Static column dynamic RAM (SCRAM) แทนหน่วยความจำแบบ DRAM เวลาการเข้าถึงข้อมูลจะอยู่ใน 40-55 ns จะเป็นการเพิ่มประสิทธิภาพของเครื่องคอมพิวเตอร์ให้มากขึ้นกว่าเดิม ก็ขอจบลงด้วยคำพูดที่ว่า โลกนี้ยังกว้างใหญ่นัก อย่ามองเฉพาะ CPU เท่านั้นครับ